BGP Instabilities and the Worms : Data to Models---Closing the Loop

David M. Nicol
Dept. of Computer Science
Dartmouth College

Other contributors

- Michael Liljenstam, ISTS
- Yougu Yuan, Dartmouth College
- BJ Premore, Dartmouth College
- Srdjan Petrovic, Dartmouth College
- Andy Ogielski, Renesys Corporation
- Jim Cowie, Renesys Corporation

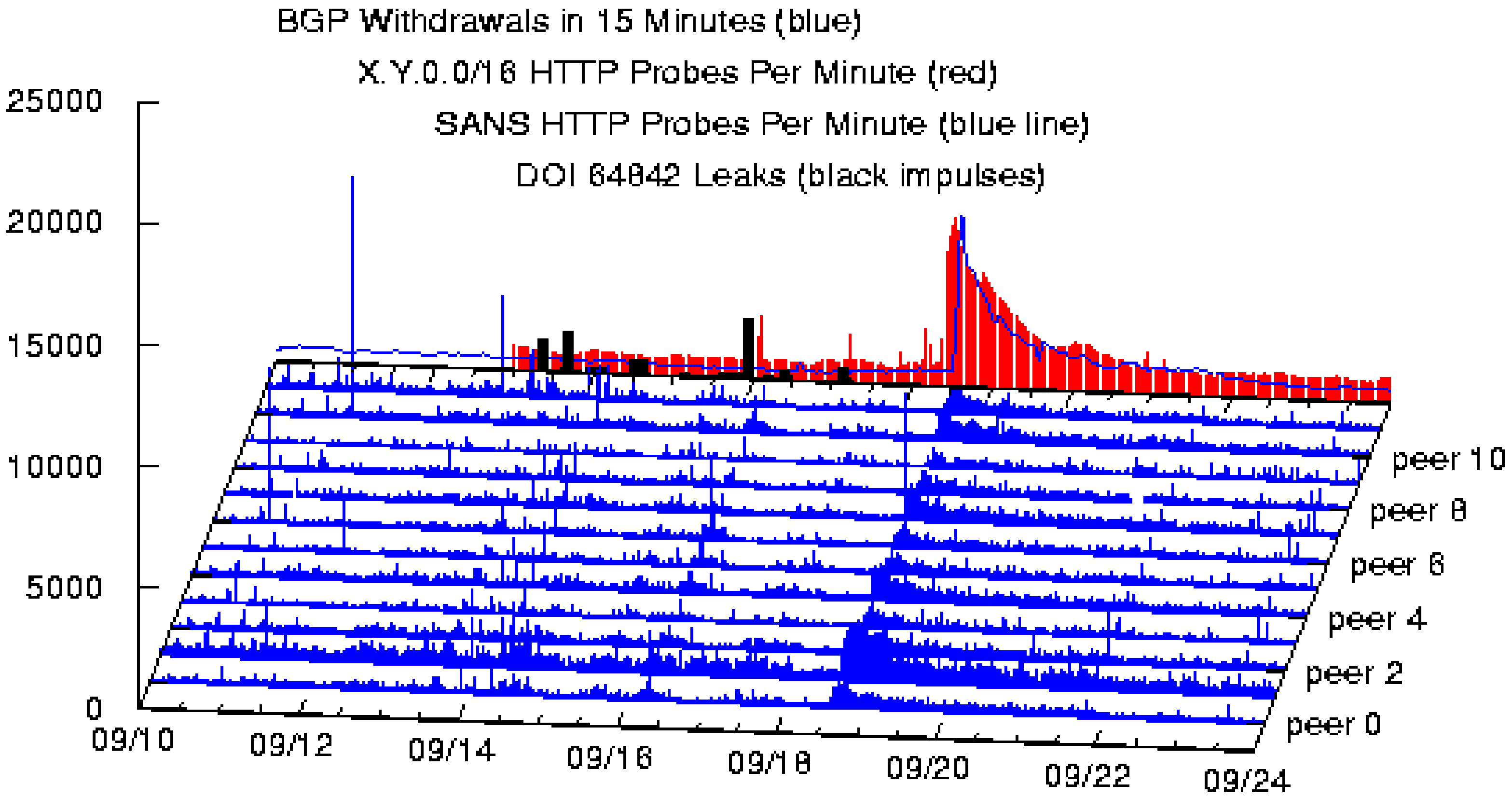
July 19, 2001

- Internet worm Code Red v2 strikes
- MS ISS servers are vulnerable
  - An infected server throws 99 threads, randomly probing IP space for victims
- In the space of 1/2 day, 350,000 MS ISS web servers become infected all over the world
- And something unexpected happened
- When nimda struck in September, ***it happened again***.

Worms caused Instability

- Measured BGP data showed that Code Red and nimba destabilized global BGP routing
- ***This was completely unexpected …***

Worms induced waves of withdrawls : nimda



See www.renesys.com/projects/bgp_instability

Closing the Loop

We’re exploring the possibility that router failure explains some of the data

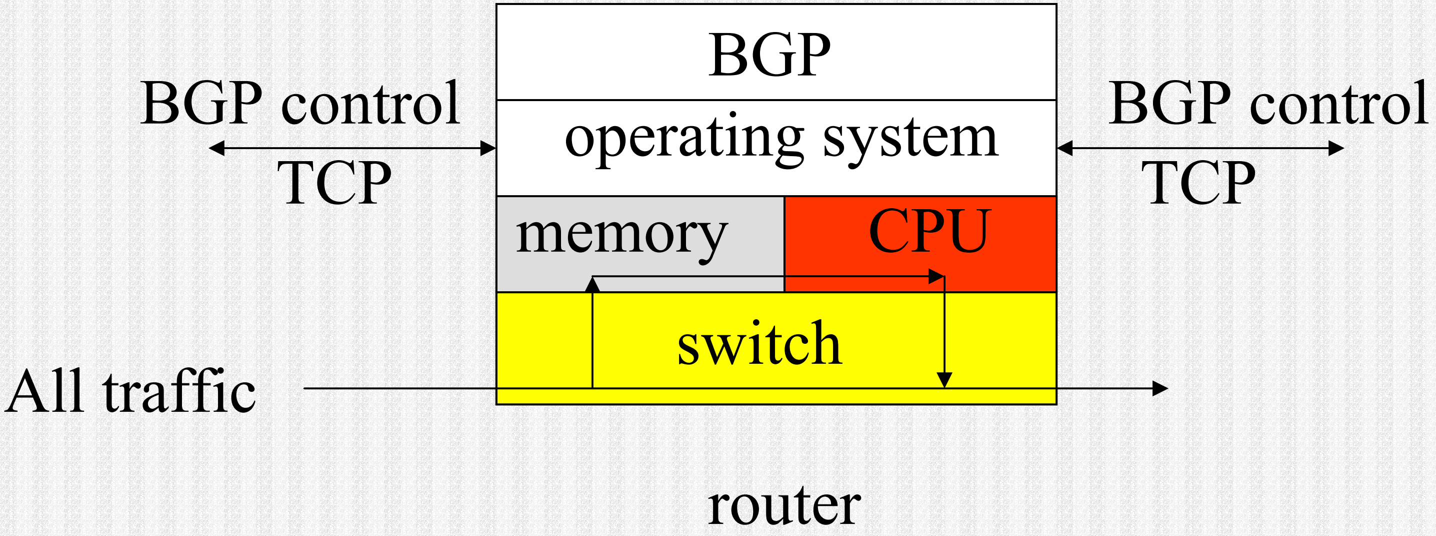
- Need to explain how routers fail because of worm traffic
- Need to explore behavioral consequences of router failure

Worm meets BGP in the Router

Facts (some anecdotal)

- Routers saw large increases in CPU utilization
- Routers are at risk under high CPU and memory loads
- IP destinations to no-where can generate ICMP packets, contributing to CPU load
- Routing is hierarchical, fast to slow
  - 1st packet of newly observed destination routed slowly
  - Random distribution of IP destinations causes many worm packets to be routed slowly (using more CPU cycles)
- Memory overload can cause router reset

Where the Action Is



When a Router drops out

- Each peer detects *non-responsive link*, tears down its BGP session across that link, announces alternative routes or withdrawals
- After several minutes, failed router finishes reboot and re-opens BGP sessions with peers
- Each peer dumps entire routing table to initialize
- In response, router announces its own paths to each unique prefix
- Whole process can take many minutes, during which time connectivity is affected

Router Model

- Maintain CPU and memory utilizations as a function of traffic and BGP demands
- Update router state every 200ms
  - Use # worm probes passed through in time-step to model CPU demand, and update utilizations
  - Use hazard-rate function to randomly sample router failure event

Worm Traffic Model

- An infected host runs 99 threads
  - Each thread manages one probe
  - A thread
    1. Sends SYN to randomly selected IP adrs
    2. Times out (21 secs.) or, receiving ACK, sends infecting HTTP GET. Randomly sampled.
    3. Repeat
- Traffic model time-stepped at 200 ms / step

Autonomous System Model

AS state is comprised of

- #infectible hosts, #infected hosts
- Array enumerating SYN launches for the next 21 seconds

Each timestep

- find total # infected hosts in reachable ASes, which gives Pr{infection}
- Sample new infections in each reachable AS
- Update router states using worm traffic this step

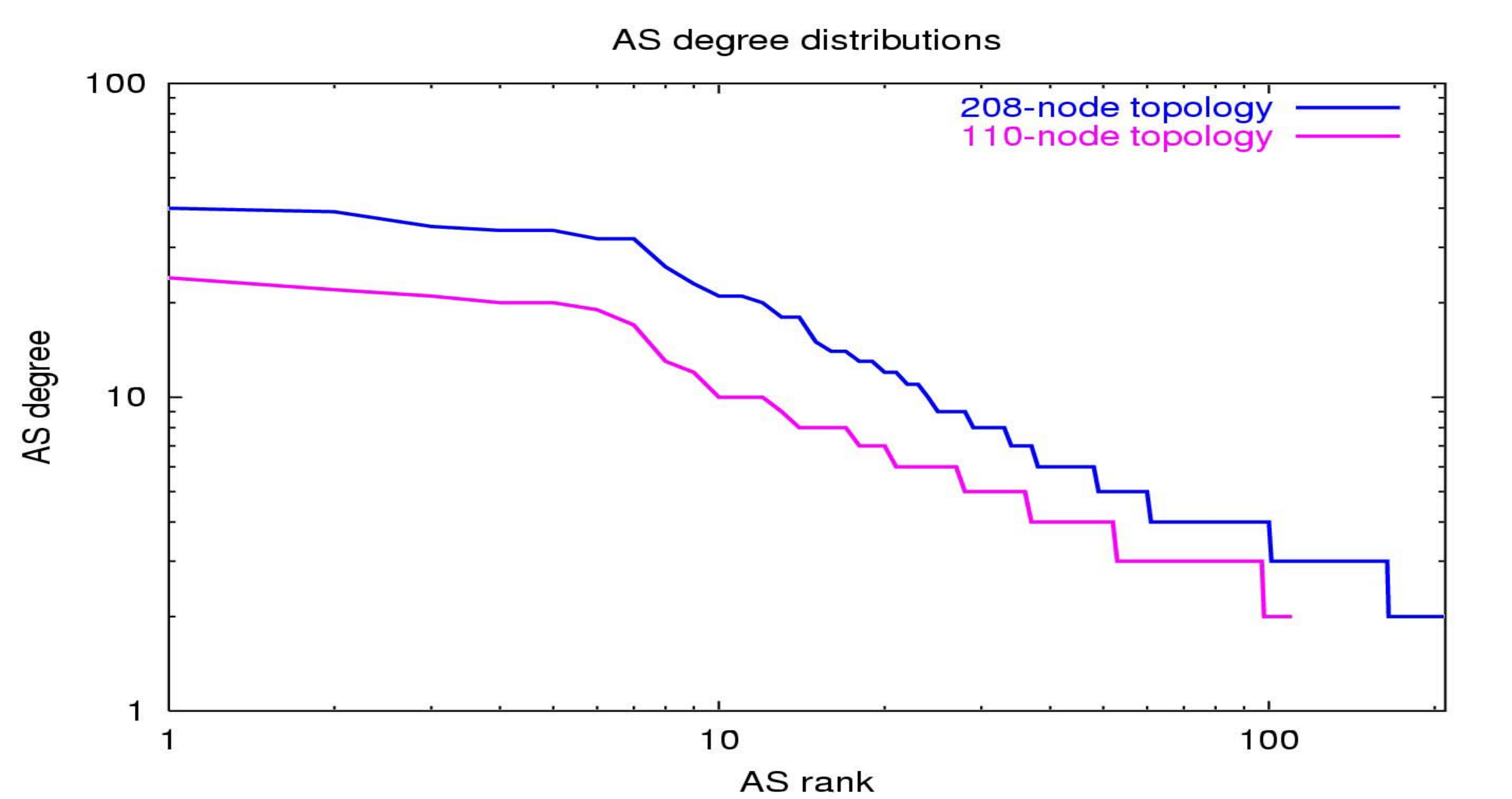
Infection growth rate closely matches measurements

AS Graph Topology

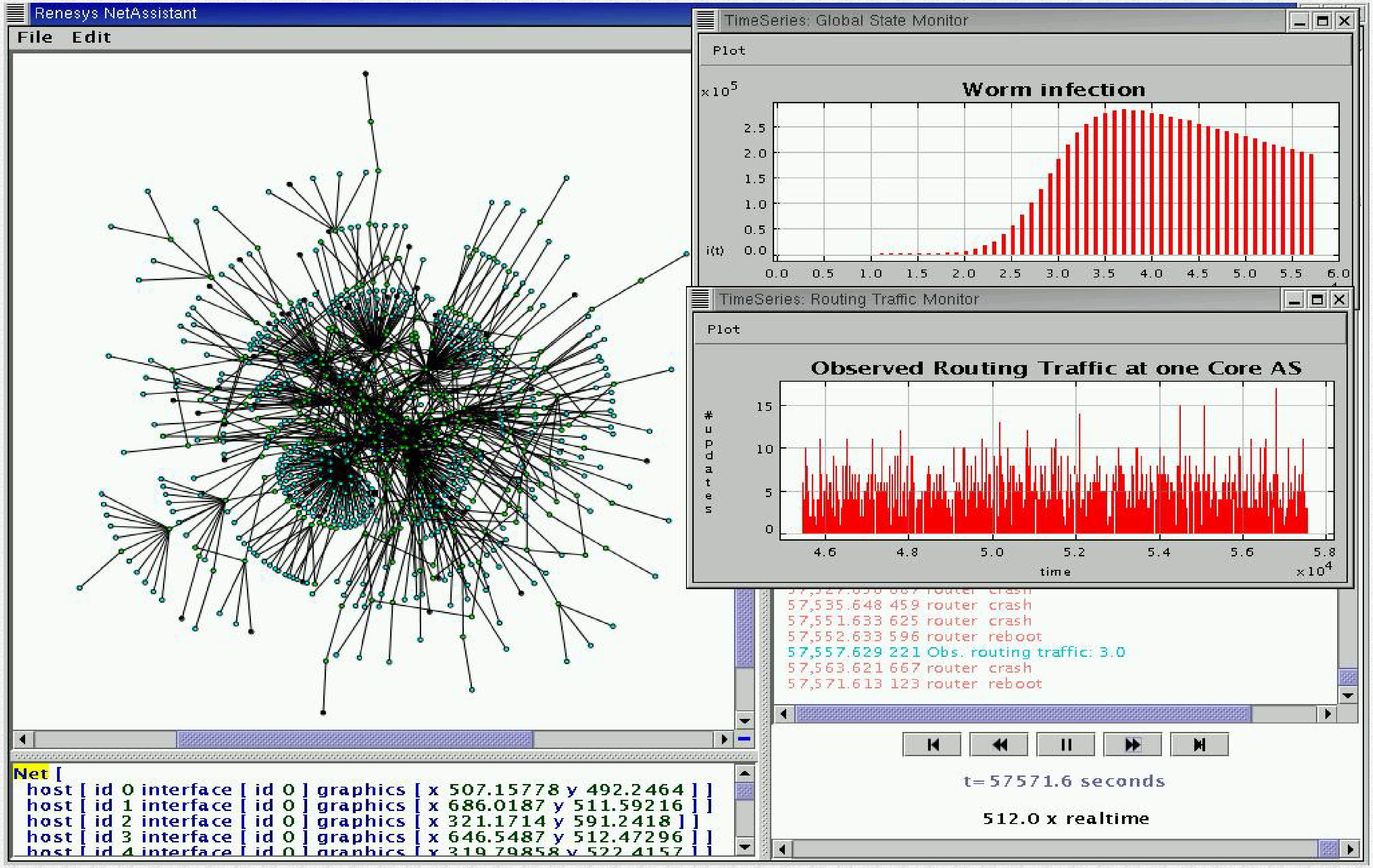
- Start with observed peering relationships, ***measured*** from Internet
  - Eliminate leaves
  - Merge nodes with low connectivity
  - Continue until desired size attained
- We study two topologies
  - 110 and 208 AS sizes (diameters 4 and 6)
  - Assumed 1 router / AS
- “Rest of the Internet” modeled with one super-AS to account for all infection traffic

Connectivity Distribution

- Large variability inherited from original topology



As Seen in Renesys NetAssistant



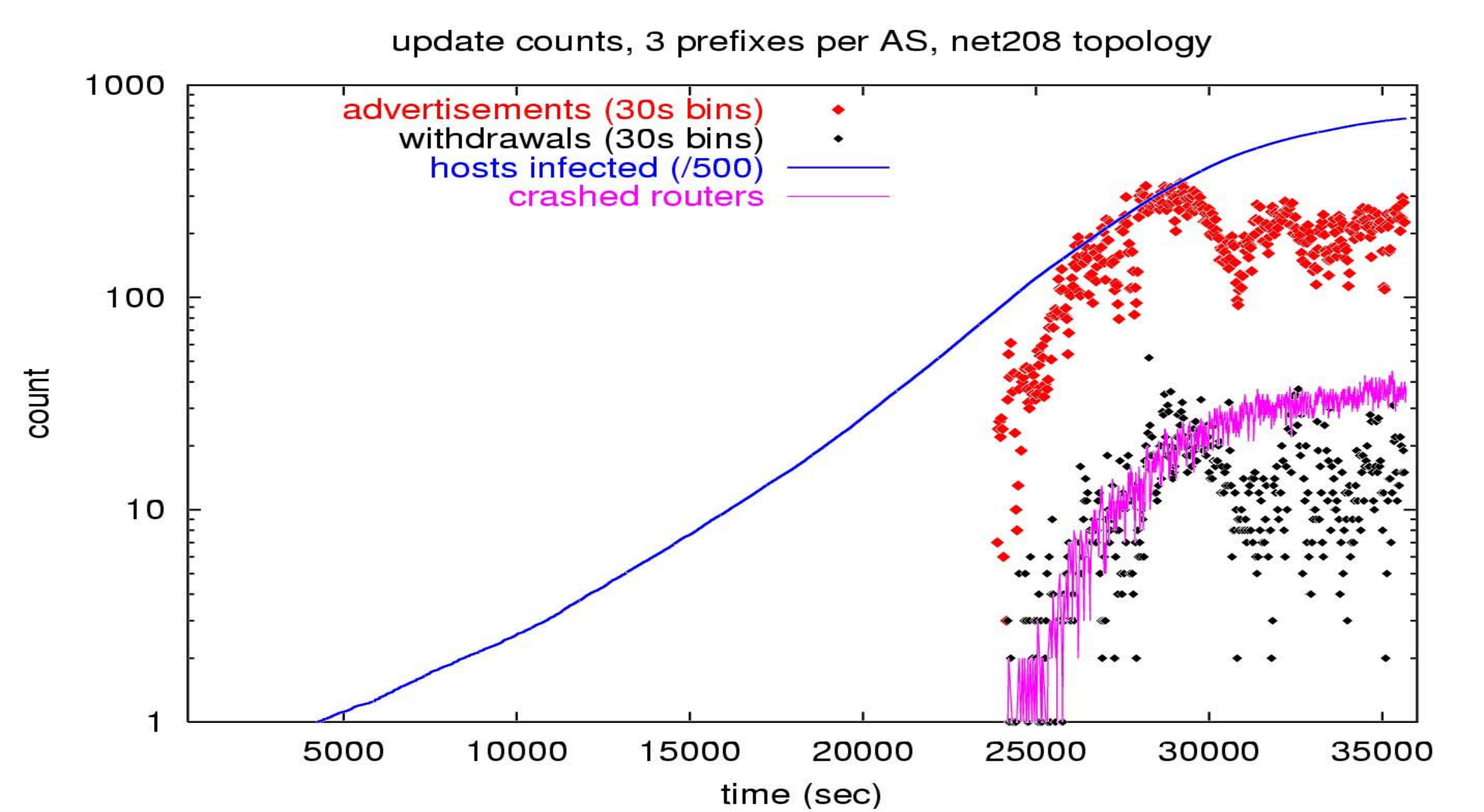
```
57,535.648 459 router crash
57,551.633 625 router crash
57,552.633 596 router reboot
57,557.629 221 Obs. routing traffic: 3.0
57,563.621 667 router crash
57,571.613 123 router reboot
```

```
Net [
  host [ id 0 interface [ id 0 ] graphics [ x 507.15778 y 492.2464 ] ]
  host [ id 1 interface [ id 0 ] graphics [ x 686.0187 y 511.59216 ] ]
  host [ id 2 interface [ id 0 ] graphics [ x 321.1714 y 591.2418 ] ]
  host [ id 3 interface [ id 0 ] graphics [ x 646.5487 y 512.47296 ] ]
  host [ id 4 interface [ id 0 ] graphics [ x 319.79858 y 522.4157 ] ]
```

Simulation Testbed : SSFNet

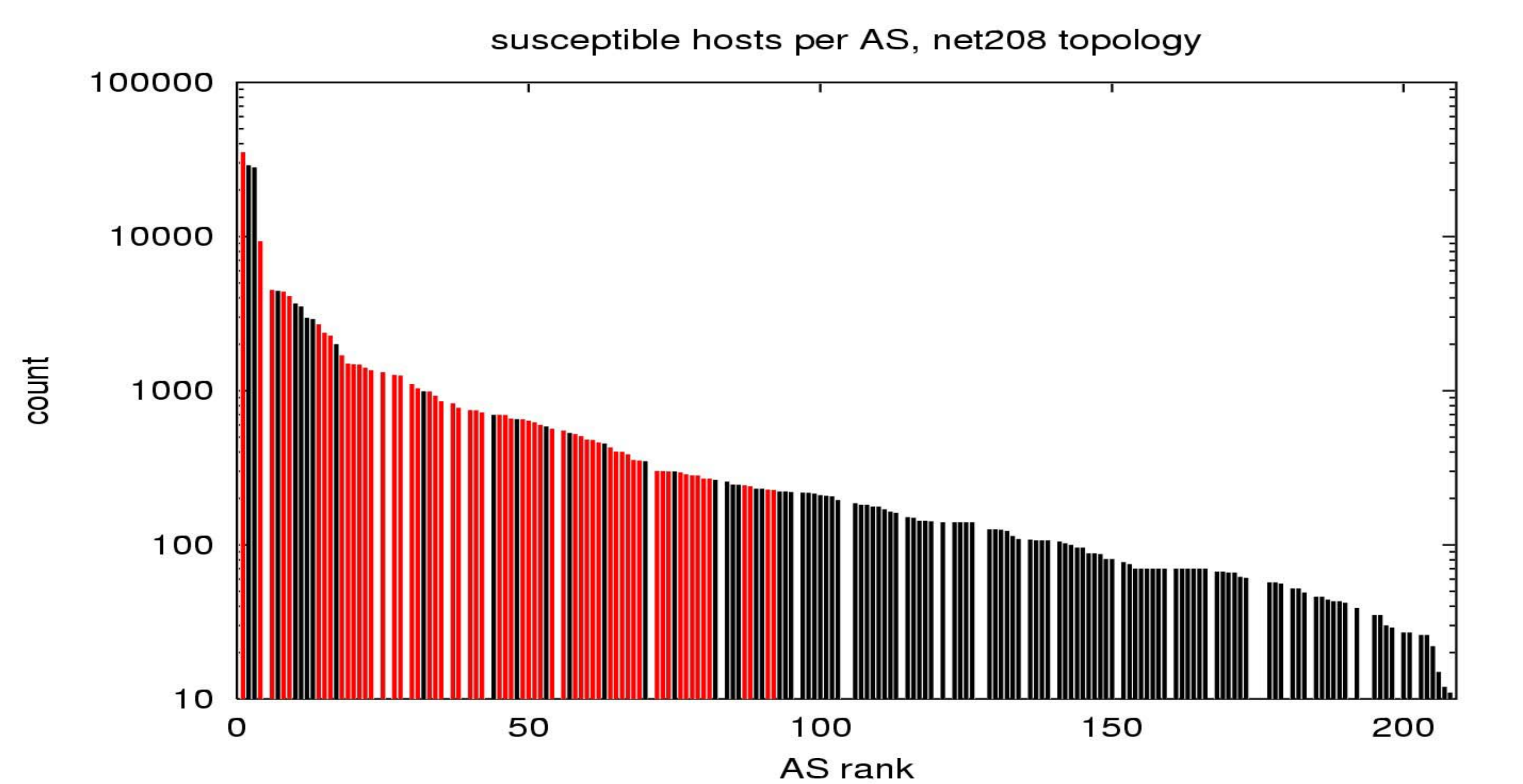
- We have already developed a very detailed model of BGP
  - Part of the Java-based SSFNet package
- New features in next SSFNet release
  - Support for session death, and reboot delay
  - Support for variable processing time for BGP messages (e.g., a function of router CPU utilization)
  - Router and worm components

Worm->crash->advertisements



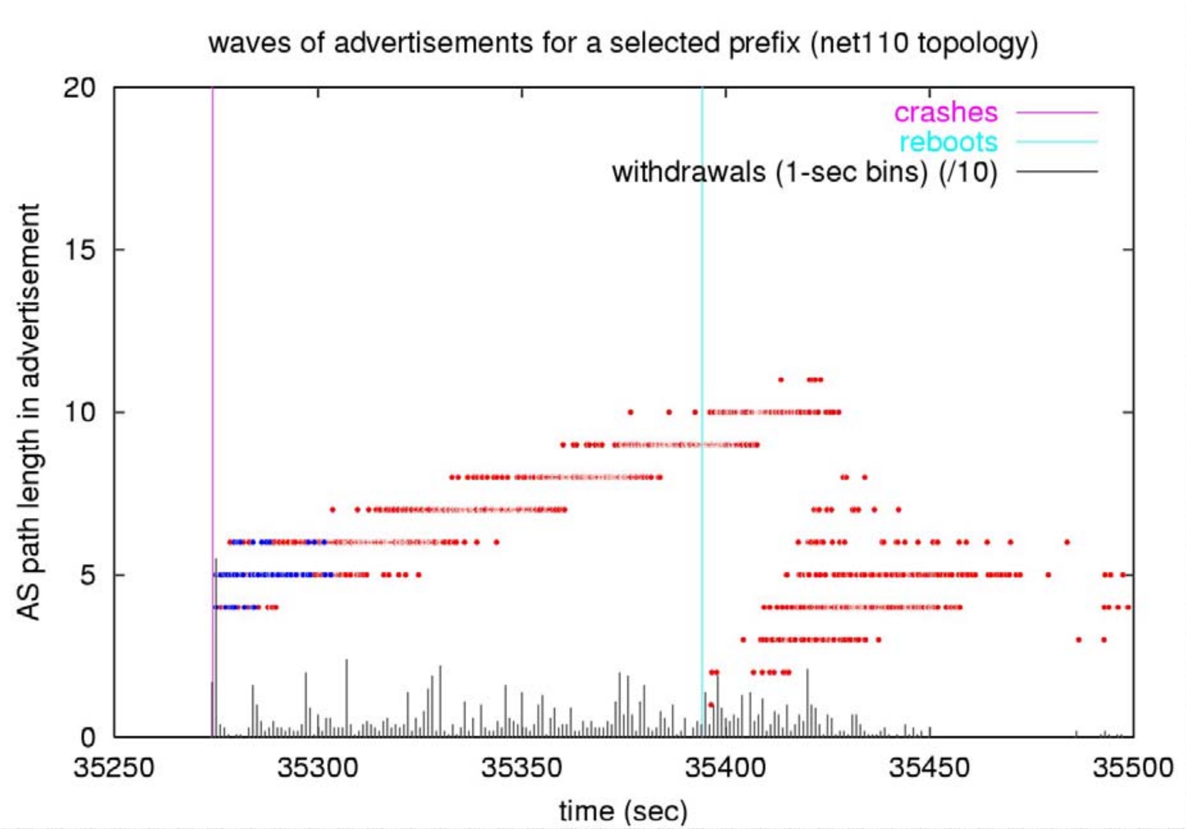
Global infection growth curve matches observations very well

Failure from within the AS



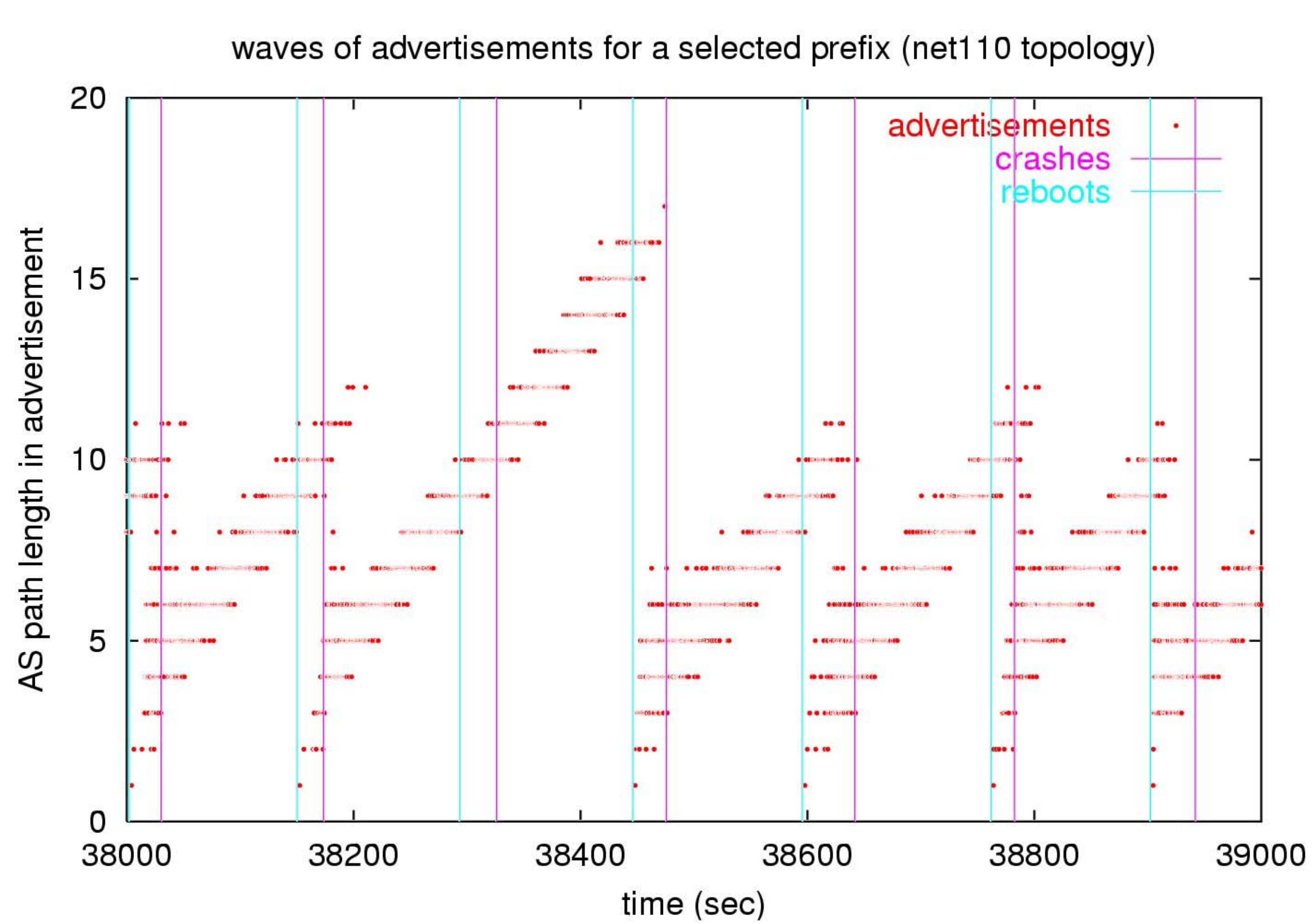
Routers fail in ASes with critical mass of susceptible hosts

Reverberations of 1 Failure



Cascading lengths due to cycling through backup routes

Overwhelmed AS keeps spreading advertisements



Conclusions

Single router failure has reverberations

- BGP cycling through backups sustains them
- BGP reacts to “link is down”, no knowledge of “router is down”

May be necessary to model BGP in detail to get this behavior, particularly if realistic filtering policies are modeled

Withdrawal waves explainable by aggregate # failed routers

Problem scale is important, even 210 node router is too small

- need problem scale to get time scales right
- need problem scale to properly assess contribution of individual router failures
- need problem scale to induce emergent interactions

We’re on to something here, but problems of complexity and scale must still be considered

More Conclusions

Problem domain challenges toolsets

- Very high memory cost
- Considerable computation cost
- Interoperability of different models at different levels of abstraction, with different time evolution paradigms

SSFNet was *designed* to support this

- We were able to respond to the need to model and explain this behavior quickly
- It is ready ***now*** to be used in other infrastructure protection and homeland defense modeling efforts